2層プラナー(計画器)の提案:自然言語理解のために(計算モデルと計算の複雑さに関する研究)

AUTHOR(S):

高橋, 英之

CITATION:

高橋, 英之. 2層プラナー(計画器)の提案:自然言語理解のために(計算モデルと計算の複雑さに関する研究). 数理解析研究所講究録 1996, 950: 146-152

ISSUE DATE:

1996-05

URL:

http://hdl.handle.net/2433/60327

数理解析研究所講究録
950巻 1996年 146-152

2層プラナー（計画器）の提案—自然言語理解のために—

Proposal of 2-Layered Planner, for Natural Language Understanding

日大理工数 高橋英之 (Hideyuki Takahashi) *

Abstract

It is known that planning is one of the key concepts of natural language understanding. My ultimate goal is to create AI Confucius through axiomatizing the Confucian Analects which is a plan document for reconstructing the society; the problem of planning is essential there.

This paper proposes the concept of two-layered planning, which is different from the ordinary hierarchical planning. The point is to separate both the goal managing task and the action series searching task of a planner; it clarifies the concept of planner. We also propose a half-planner which searches a plan, given an ordered sequence of goals, and a quarter-planner, and a plan-checker, which are near the concept of tow-layered planner.

Besides, we discuss the relation between Schank's PAM using 2-term relations and the usual planner using 3-term relations.

1 要約

自然言語理解の一つの要点は、プラニングにあることが認識されている [Okada91a],[Okada91b], [Allen94]。筆者の研究目標である「論語の公理化 ⇒ AI 孔子ソフト作成へ」においても、目的とそれに向かってのプラニングが、本質的である [Shiraishi95, pp.7-9, pp.230-231]。

階層プラニングの研究もなされているが、本講演においては、いわゆる多層ではなくて、2層のプラナーを提案する。これは、目標を管理する上層プラナーと、個々の目標実現のための行為系列（プラン）を見つける下層プラナーからなり、Prolog で比較的容易にプログラミングできる。これを、planner_2nd と呼ぶ。2層あれば、好きなだけの多層化もできる。目標管理と行為系列探索を分けることで、プラナーが概念的にすっきりする。この概念的明確化が成果である。

さらに、2層プラナーに近いものとして、目標の時間的順序を与えて計画させる half_planner や、途中の一里塚となる目標系列を与えて詳細を求めさせる quarter_planner や、プランを与えて正しいか否かを確かめさせる plan_checker なども、一連のソフトとして提案ないし位置付けできる。

プラニングに関しては Schank らが PAM というソフトを提案したが、これは2項関係を扱い、状態オペレータとしての行為を3項関係で扱う通常のプラナーとは、少なくとも見かけが異なっている。両者の関係如何が興味ある点であるが、大体のところ、PAM の2項関係は、通常の3項関係を、前の2つと後ろの2つとに分割したものであることが分かった。

自然言語研究（LA の L）の基本的方法は、演繹ではなくソフト作りにあると思われる。

2 導入あるいはモティベーション

[Schank81] は自然言語研究において画期的であった。その理由は次の2点にある。

（A） これが文法（構文）でなく、「意味」をしっかり捉えていたこと。

* E-Mail: tkh@math.cst.nihon-u.ac.jp

（B） 1つの文でなく、「複数の文のつながり」を扱っていたこと。
(A) 前者については、格形式と、「基本行為説」とを結合した、彼らがCD (Conceptual Dependency) と呼ぶ、例えば、
“John gives Mary a book” に対しては、

(ATRANS (ACTOR (PERSON (NAME (JOHN)))) (OBJECT(BOOK))
(TO (PERSON (NAME (MARY))))
(FROM (PERSON (NAME (JOHN)))))

のような形式を用いて、文の意味を表す。これが意味だというのは、これを用いて “意味に関する計算” ができるからである。英語からCD形式への変換（これは難）と、その逆の、CDから英語への変換（これは比較的簡単）の、Lispによるソフトも示されている。
(B) 後者に関して、一般に複数の文のつながりについては次のようなものが考えられよう。
(1) フローチャート、あるいはSchankらの呼び方ではscript。これは人が初めから筋書きを与えるものである。変数入りにして筋の「パターン」を扱えるようにする。
(2) 数学の証明など演繹論理の文。一二の以前の文から推論規則により新しい文が生成される。
(3) プラニングによる行為の系列。行為と行為の間に、状態が挟まれるのが普通。これは（状態1、行為、状態2）という3項関係を人があらかじめ与えて、

（現在の状態、行為1、状態2、行為2、状態3、...、行為n、目標状態）

と、行為による状態変換を繰り返し、目前の状態から目標の状態にまでいたろうとする。つまり現在と目標との間をつなぐ行為系列を、コンピュータに探索させるものである。
(4) 事実の自然な推移による状態遷移の系列。これには、事実とそれに対する或る価値観による価値評価、そしてそれによる反応、といったつながりも含まれている。基本は、事実–事実の2項関係をもとに、因果的連鎖を作るものである。
(5) これらの組合せ。例えば、上記(1)の中に(2)を含ませ、また(3)の中に(2)を含ませる、等。

Schankらが扱うのは(1)(3)(4)(5)である。全体の中で、プラニングの占める要素が非常に大きい。言葉は、自他の人間の行為を記述するものが多く、人間の行為は意図・目的をもって計画したものが多いからである。こうして、自然言語理解にとってプラニングが本質的となる。

筆者はここ数年、「論語」の公理化⇒ AI孔子ソフト作成へ、という研究目標をもって関連ソフト作成に努力しているが、その研究が可能であると確信したのは、[Schank81] を知ったからに他ならない。「論語」は孔子の、社会復興計画の計画書である（一つの解釈として）。論語という自然言語の文書において働いている中心的な思考は、プラニングである。これが、筆者がプラニングを詳しく調べなければならないと考えた動機である。

3 通常のプラナー、本稿の選択

Prologで書かれた標準的なプラナーは、例えば [Bratko90] や [Sterling94] にある。これは探索の一種である。探索 (search) とは方法の名前であり、プラニングとは問題の名前である。よく扱われ、本稿でも扱う具体的問題は、「猿–バナナ」問題と、積木問題である。

比喩をいえば、物理学では、微分方程式で書かれた法則を積分することが問題であるが、プラニングは、3項関係（状態1、行為、状態2）で書かれた微視的関係を、積分して、初期状態と目標状態を結ぶことが問題である。基本のところに、「行為とは状態を変えるオペレータである」という観点がある。これは、いわゆる言語学の方でも認められている見方である [Ikegami75] 。これですべての動詞が把握できるわけではないが、まず初めのとっかかりとしては有力な考え方である。

上記2つの問題に対する “状態変換オペレータとしての行為” の記述を書こう。もちろんこれは

現実をモデル化・オモチャ化したものであって、現実のごく一部を反映するものにすぎない。
(1)「猿–バナナ」問題：これには次の4つの行為がある。
GRASP：適切な場所で掴むとバナナを持てる。/ CLIMB：椅子の所で実行すると高い位置に行く。
PUSH ：適切な場所で押すと椅子と本人が移動する。/ WALK ：場所を移動する。

PUSH の場合の行為前状態と行為後状態を書くと、次のようになる。

```
move( [at(monkey,(P1,low)), at(chair,(P1,low))],  '****PUSH****'( P1, P2),
   [at(monkey,(P1,low)) => at(monkey,(P2,low)),  at(chair,(P1,low)) => at(chair,(P2,low))] ):-
          place(P1), place(P2), P1 \= P2.
place(door).  place(window).  place(middle).  place(corner).
final_state( [has(+,monkey,banana)] ).
initial_state( [at(banana,(middle,high)), at(chair,(window,low)),
                                  at(monkey,(door,low)), has(-,monkey,banana)] ).
```

(2) 積木問題： TO_PLACE と TO_BLOCK の2行為。詳細は略。

世界は (1) では部屋と猿・椅子・バナナからなり、場所は水平位置 H と垂直位置 V で表される。H は、door, window, corner, middle のどれかの値、V は low と high のどちらかの値を取る。(2) の世界は3つの場所 p,q,r と、その上に載る積木 a,b,c である。

状態記述の形式には選択の余地があるが、ここでは、状態＝命題の集合、とする。命題とは例えば、「猿が場所 (H,V) に居る」—at(monkey,(H,V))—というようなもので、これは、より詳細に表すには、格形式—[be, actor:monkey, loc:(H,V)]—を採用しなければならなくなる。

move(状態1 , 行為, 状態2) において、状態1は、トータルな状態のうちの必要条件のみを記している。状態2の書き方にも種々あるが、ここでは、状態1の命題がどう変化するかを記述する形式をとる。+A は、命題 A が付け加わること、–A は、命題 A が除去される（不成立となる）ことを表す。A⇒ B は、(–A,+B) と同値である。

探索には、縦型探索、横型探索、最良探索があるが、ここでは最良探索をとる。最良選択では、状態（とパス path）に対する評価関数をどうとるかが本質的であり、評価関数の取り方次第で、手数をプラス評価すれば縦型、マイナス評価すれば横型、というふうに、すべてを含んでしまい、しかも良い評価関数は結局、解が分かってから分かる、といった具合であるが、その設定はユーザの責任、ということにして、ここでは最良選択をとる。

探索における推論には、(1) 前向き推論 initial-to-final(i2f と略記)、および (2) 後ろ向き推論 final-to-initial(f2i と略記) の両者がある。i2f は正確ではあるが、時間がかかる傾向がある。f2i は、時間が速いが、遡ることがうまく行かないことがある。我々は、このどちらでも、ユーザの指定で選べることにする。例えば、猿–バナナ問題で f2i とすると、「椅子まで歩き、椅子をバナナの下まで押し、椅子に登り、バナナを掴む」という解が得られる。

4　2層プラナーとその系統のソフト

[Sterling94] のプラナーは、目標状態を一直線に求めようとするものであるが、現在のプラニングは、論理としては [Chapman87] で完成した、と見られている。これは大まかなところ、目標状態をなす命題の各々を個々に目標とみなし、それら目標を順に実現してゆく、一旦実現した目標がもし破壊される恐れがあるなら、回復が保証されねばならない、と主張している。[Bratko90] は、おおむねその論理にかなったアルゴリズムを述べている。ただし、論理がどうであるかと、プラナーの構造がどうであるかは、別の問題である。

これらを見るとき、我々は次の認識をもつ。即ち、プラナーは2つの働きをする。

(1) 目標を与えられて、それを実現する行為系列（即ちプラン）を見出すこと。

(2) 目標の管理、つまり目標状態は一般に複数の命題（conjunctive goals）で構成されるが、それら目標をどのような順番で実現してゆくか、実現した目標が破壊されることはないか、などの管理を司ること。

これら2つの機能がプラナーに持たされているが、後者の機能は十分意識的に把握されているとは思われない。ここで我々は、これら2つの機能を分離することを考える。つまり、行為系列を見つける役割と、目標を管理する役割とを、分けて別のソフトに担わせることを考えることにする。管理を司るプログラムを solve_best_2nd とし、それが設定した初期状態と目標状態に対して行為系列を見出すのは今まで通り solve_best とし、2つを合体させたファイルを planning_2nd と名付ける。管理者が2層目、被管理者が1層目をなし、前者が後者を従える、という構成になる。

こうして2層にするが、似たプラナーが2階建てになっているのである。

(1) 1層目のプラナーは、初期状態とゴール状態を（2層目に）与えられて、そこへ至る動作系列を求める通常のもの。

(2) 2層目のプラナーは、ゴール＝命題（集合）を管理する。その命題を1個ずつ取り出して、1番目のプラナーを使用人として使って、1命題を達成する動作系列を求めさせる。

planning_2nd　2層目：目標管理プログラム planner_2nd.　述語は solve_best_2nd。
　　1層目：行為探索プログラム。planner_1st. 述語は solve_best。

一度 達成した Goal に関しては、以後 そのどの一つをも損なわないようにする（損なっていないことを確認する）ことが重要である。1個の Goal の達成には既成の Prolog プログラム best_first_search (solve_best) を使い、複数の Goal の実現順序の発見には、新たに作った第2レベルの best_first_search (solve_best_2nd) を使う。1、2階とも、内容的には似たプログラムとなっている。

ほとんど同じ構造のプラナーが2層になっており、その各々は、いくつかの選択肢を独立に選ぶことができる。例えば、i2f (前向き推論) か f2i (後ろ向き推論) かである。経験的には、1層目を後ろ向き推論に、2層目を前向き推論にすると良い。評価関数も2つの層で別にする。

こうして、プラニングのやり方を2階建てにした。1階での単位は動作であり、2階での単位は目標である。この考え方でプラニングが整理した形で理解できるようになった。実行時間はかえってかかるようになる（筆者の Sun Spark IPX で数分、これは多分に評価関数に左右される）、が、概念的にはすっきりする。長期的にはそれが大事だと考える。猿-バナナの問題の出力例を示す。

```
[at(banana,(middle , high)), at(chair,(window , low)),
                                    at(monkey,(door , low)), has(-,monkey,banana)]
:=> get_GOAL(has(+,monkey,banana))/[get_GOAL(has(+,monkey,banana))/
        [****WALK****(door,window), ****PUSH****(window,middle),
            ****CLIMB_UP****(middle), ****GRASP****(middle,high)]]
=> [at(chair,(middle , low)), at(monkey,(middle , high)),  has(+,monkey,banana)]
:=> get_GOAL(at(chair,(corner , low)))/
        [****CLIMB_DOWN****(middle), ****PUSH****(middle,corner)]
=> [has(+,monkey,banana), at(monkey,(corner , low)), at(chair,(corner , low))]
:=> get_GOAL(at(monkey,(window , low)))/[****WALK****(corner,window)]
=> [has(+,monkey,banana), at(chair,(corner , low)), at(monkey,(window , low))]
```

目標状態は [at(monkey, (window , low)), at(chair, (corner , low)), has(+, monkey, banana)] であるが、その実現の順序、即ち、まず has(+, monkey,banana)、次に at(chair, (corner , low

))、最後に at(monkey, (window , low)) という順番は、2層目のプラナーが見出したものである。状態の各命題を個々の目標とはせずに、まとめて扱う [Sterling94] のやり方では、上記の問題くらいは、わけなく解けるが、各命題を目標とする本稿の方法は、それとは別の興味を追求する。

get_GOAL(has(+, monkey, banana)) のところに見られるように, 4つ以上の行為系列は、ひとまとめに括れるようにしてある。このメカニズムにより、ボトムアップ的に、行為を "マクロ化" してゆくことができる。これは一種の階層化である。以後は、この「マクロ化行為」を単位として用いてプラニングができる。

なお、find_all_moves_2nd の中で find_all(Ms, solve(Ms)) を使っているため、複数の解が次々に求まる。例えば、猿が歩いてまたは椅子を押して不要な寄り道をしたり、椅子に上がっては降りる、などの遊びをする解である。1層目の solve(Ms) が見つける別解は、途中の寄り道や遊びを含めれば、一般に無限個ある。それをどこで切るか、が一つの選択肢である。個数を何個までとすることも考えられるが、所用時間がひどく不揃いとなる。むしろ、時間を一定として切った方がよい。

積木の場合、例えば初期状態 [a on b, b on c, c on p] から目標状態 [a on b, b on c, c on q] に行こうとするが、 c on q, b on c, a on b という実現の順番を、2層目のプラナーが見出すことができる。

以上2例における目標実現の順序には、はっきりした特徴がある。目標を A, B, C とするとき
「A の実現は B、C を乱す。 B の実現は C を乱す。 C の実現は何も乱さない。」

目標間にこういう関係があるとき、実現の時間的順番を、まず A、次に B、最後に C の順にすることは、確かに理にかなっている。move(状態1 , 行為, 状態2) の各定義をコンピュータに分析させて、上のような目標間の優先順位を見つけさせることができるなら、プラニングの問題はかなり簡単化されるはずだが、筆者はこの問題はまだやっていない。

さて、以上の2層プラナーの系列上に位置して考えられるいくつかの Prolog プログラムを作成した。これらは特に目新しいものではないであろうが、まとめて一群のものものとして捉えられるという意味で、併せて考慮した。詳細は略す。

(1) 半プラナー half_planner : ゴールたちの、実現の時間的順番を入力として与えられ、その具体的な動作系列を通常のプラナーを使って求めさせる。

(2) 四半プラナー quarter_planner : これは、一里塚のように目指すべきゴール—但しそれらは行動の中途で現われる命題である—の系列を与えられて、それらを次々に追求することにより、ついに最終状態に到着するものである。

(3) プラン・チェッカー plan_checker : プラン、即ち行為の時間的系列を与えられて、それが本当にゴールを達成できることをチェックする。

5 PAM—Schank 流のプラニング

[Schank81] の PAM (Planning Applier Mechanism) は、プラニングによって談話を理解しようとするものである。簡単な例としては、「ウィラは空腹だった。彼女はミシュラン・ガイドをつかんだ。彼女は車に乗り込んだ」という一連の文を、我々人間が理解するのと同じ背景をもってコンピュータが理解できるようにしようとする。そこでは断片的なゴール–プランの対をもとに、
「レストランのガイド・ブックをつかむ ⇒ レストランのガイド・ブックを読む ⇒ レストランを知る ⇒ 車に乗る ⇒ 車で行く ⇒ レストランで食事 ⇒ 満腹という目標設定 ⇒ 空腹という動機」
という一連の前向き推論をさせることを試みている。

本節で問題にするのは次のことである：プラニングとは、通常のプラナーにおいては、行為＝「状態を変化させるオペレータ」、状態＝「行為によって変化を受けるオペランド」という観点で、

move(State1, Action, State2)、つまり、(状態１、動作、状態２)

を用いて、初期状態を目標状態へと導く動作系列を求めることであり、要するに“３項関係”を扱う。

しかるに、Schank & Riesbeck は、PAM その他で、プラニングを、もっぱら二つの CD のペアによって記述して行こうとする。CD (Conceptual Dependency) のペアとは、(結合子、CD １、CD ２) という形の、要するに“２項関係”である。

以上二つの観点、即ち、(状態１、動作、状態２) という観点 (３項関係)、と、結合関係 (CD １、CD ２) という観点 (２項関係) という両者は、一体どんな関係にあるのか。それが疑問である。

[Schank81, pp.192-3] にある計１３個の CD ペアについて、詳細に検討した結果、得られた結論を述べると—

まず、CD には実は２種類ある。同じ CD という名前でも、「状態を表す CD」と「行為 (動作) を表す CD」とがある。そして大まかに言って、Schank & Riesbeck 流の２項形式は、通常のプラニングの３項形式を、２つに分けたものである。つまり、

(状態１、動作、状態２) を、

(状態２、動作) および (動作、状態１)

と２つに分けたものである! 但し、(状態１、状態２) というペアもあるので、注意が必要である。

筆者はこれに関連して、次の２つの Prolog プログラムを作成した。

(１) pamrule2move : これは、２項関係を３項関係に変換する。すなわち、[Schank81] とほぼ同じ CD ペアを与えられて、通常のプラニングのような３項関係 move (条件１、行為 a、条件２) を生成するものである。これにより、PAM と通常のプラニングの橋渡しができる。

(２) move2plan : これは、上記 (１) の結果を受けて、３項関係の行為オペレータを辿って、通常のプラニングを行なうものである。これにより、PAM にあるのと同じプランを生成できた。

(３) TALE-SPIN にプラニングを入れたタイプのプログラム。これを少し述べる。

[Schank81] は物語生成ソフト TALE-SPIN をも提出している。これは、ユーザが舞台設定・状況設定をして、登場人物の一人が、空腹である、ないし、渇いている、などの問題を持ち、他の人物と「取引」をするなどして例えば食物の場所を知り、無事見つけて食べて、めでたしめでたしとなって終わるものである。[Schank81] の Micro TALE-SPIN は、中でプラニングを使ってはいないけれども、一般には当然、物語はプラニングを含む [Okada91b] 。筆者はこの TALE-SPIN を消化吸収すべく、下記３つの Prolog プログラムを順に作成した。一つの関心は TALE-SPIN において

「状態変換オペレータとしての行為」と、「状態」とが、どのように扱われているか

という点である。TALE-SPIN は、この点に関して十分には意識的ではない。

TALE-SPIN は「状態」として、物の位置などの物理的な状態だけでなく、人物の心的状態—その知識、目標、デモン等—をも含んでおり、その点できわめて興味深いシステムである。

(a) tale_spin : TALE-SPIN をほぼ忠実に Lisp から Prolog へ翻訳したプログラムである。

(b) tale_spin_without_cycle : TALE-SPIN では、

状態変化 ⇒ 人物の反応 ⇒ 状態変化 ⇒ それによるまた反応

というふうに一連の事態が続くのに対して、その各フェーズを１サイクルとしてサイクルを回転させる。これは世界の運行を単純にする一方、反応のシーケンスが見づらくなる原因となっている。

そのため、行為とその結果の状態変化、という点に注意しながら、サイクルをほぼ全廃して、行為から状態変化へ、また、状態変化から行為へ、というつながりが、(２項関係ながらも) サイクルを経由せずに直接生じるように改めた。かつ、各行為の状態変換オペレータの性格を明確化した。これにより、プログラムの構造が大幅に単純化された。

(c) tale_spin_with_planning : 上記 (２) のプログラムの中に可能なかぎりプラニングを導入した。ただ、ここでの状態変化は、状態変化そのものを記述するのではなく、「変化した状態に対し

て或る述語が成立することを要請する」形となっている。これでシステムのかなりの部分がプラニングで扱えるようになった。しかし、persuade（説得、具体的には ask–依頼、 bargain–取り引き、threat–脅し）の筋書きを、プラニングによって生み出すことは、できないように思われる。これらは、そういう方法があるのだとして頭ごなしに教えてやらなくてはならないように思われる（もしかすると生成できるのかも知れない）。

6 まとめ

筆者がプラニングに関して理解できたところを述べた。これはもともとは、[Schank81] に衝撃を受けたことに端を発したもので、自然言語理解にプラニングが重要であるということで、まずはプラニングそのものの概念的明確化をめざした。プラナーの機能が、目標の管理、そして行為系列の探索、というふうに二重になっていることに着目して、その両者を分けて、2層建てのプラナーのソフト（とその系統）を作成してみた。

次に、[Schank81] の2項関係のプラナーが通常の3項関係のプラナーと見かけが異なることから、両者の関係をしらべ、両者を橋わたしするようなソフトをいくつか作成した。これらを通じて、行為とは状態変換オペレータである、という既知の観点は、あくまでも有効であった。

個別的なソフトを作るだけでなく、今後、自然言語研究用ソフトウェアの社会的基盤（インフラ）を作ってゆくことが望まれる。

参考文献

[Allen94] J. Allen, *Natural Language Understanding, 2nd Ed.*, The Benjamin/Cummings Pub. Co., Inc., 1994.

[Bratko90] I. Bratko, *PROLOG–Programming for Artificial Intelligence*, Addison-Wesley Pub. Co., 1990. (一部翻訳あり).

[Chapman87] D. Chapman, "Planning for Conjunctive Goals", *Artificial Intelligence*,vol. 32, pp.333–377, 1987.

[Ikegami75] 池上嘉彦、「意味論—意味構造の分析と記述」、大修館書店、1975、1993（7版）.

[Okada91a] 岡田直之、「自然言語処理入門」、共立出版、1991.

[Okada91b] 岡田直之、「語の概念の表現と蓄積」、電子情報通信学会、1991.

[Schank81] R. C. Schank, C. K. Riesbeck, *Inside Computer Understanding*, Lawrence Erlbaum Associates,Inc., 1981.
石崎他訳、「自然言語理解入門—LISP で書いた5つの知的プログラム」、星雲社、1986.

[Shiraishi95] 白石明彦、「人工生命とは何か」、丸善ライブラリー、1995.

[Sterling94] L. Sterling, E. Shapiro, *The Art of Prolog, 2nd Ed.*, The MIT Press, 1986（初版）, 1994.
松田訳、「Prolog の技芸」、共立出版、1988.